GNUstep

Conceptual Architecture

Group 18
URL: https://youtu.be/rg_Hr1Hfa6o

Group Distribution

- **Lixing Yang**

Group Leader & Abstraction & Introduction and Overview & Use Cases & Lessons Learned

- **Chi Ma**

Presenter & Concurrency In GNUstep & PowerPoint & Video & Lessons Learned

- **Tiantian Sang**

Presenter & Division of Responsibilities & PowerPoint & Video & Lessons Learned

- **Nick He**

Derivation Process & Control and Data Flow & Lessons Learned

- **Dunyi Xie**

Interacting parts & Use Cases & Lessons Learned

- **Zhiming Jin**

Software Development Stages & Lessons Learned

Abstract

GNUstep is a robust, **open-source** framework designed for developing **cross-platform applications** that adhere to the OpenStep API, a specification originating from NeXT and later influencing Apple’s Cocoa API.

Introduction and Overview

The report is structured as follows:

Section 1: Abstract

↓

Section 2: Introduction & Overview

↓

Section 3: Derivation Process

↓

Section 4: Interacting Parts

↓

Section 5: Software Development Stages

↓

Section 6: Concurrency

Section 7: Control and Data Flow

↓

Section 8: Use cases

↓

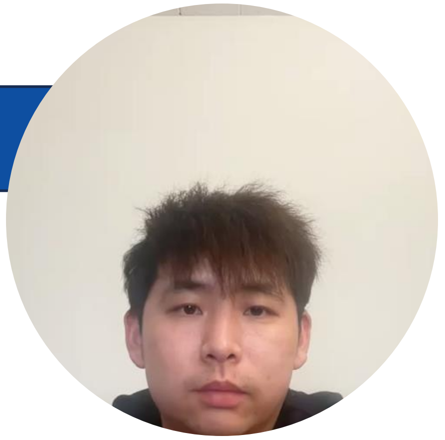
Section 9: Division of Responsibilities

↓

Section 10: Lessons Learned

↓

Section 11: References

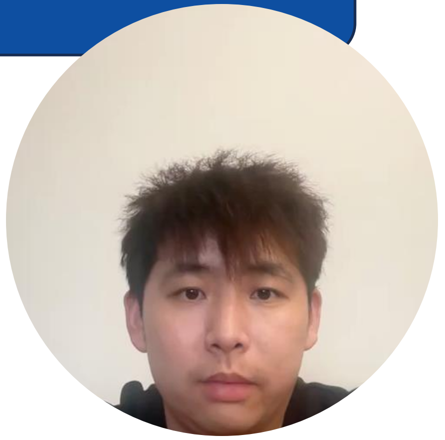


Derivation Process

Our study of GNUstep shows that its architecture is **layered** and **object-oriented**

Layered Architecture

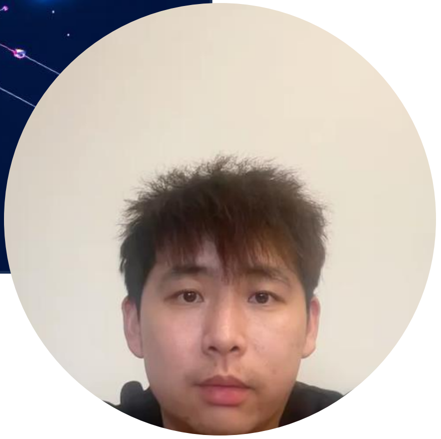
Object-oriented Architecture:



Interacting Parts

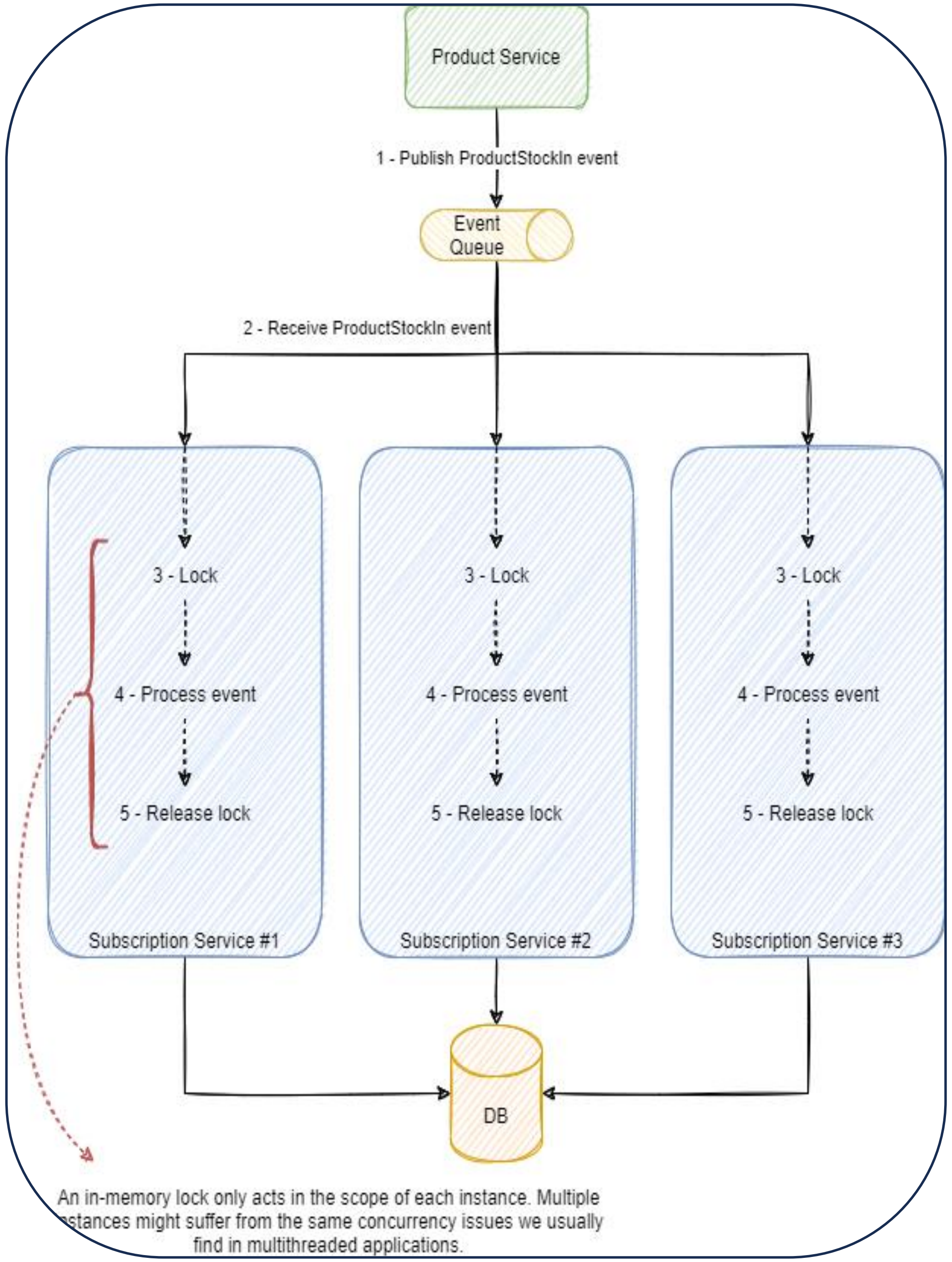
Components:

- The Foundation library
- The gnustep-gui
- GNUstep-back
- GNUstep Make



Software Development Stages

- Early Development (1993-1994)
- OpenStep Implementation (1994-1995)
- GNUstep Base and GUI (1995-2000)
- Stabilization and Expansion (2000-2
- Modern GNUstep (2010-Present)


Product Service
1 - Publish ProductStockIn event
Event Queue
2 - Receive ProductStockIn event
3 - Lock
4 - Process event
5 - Release lock
Subscription Service #1
Subscription Service #2
Subscription Service #3
DB
An in-memory lock only acts in the scope of each instance. Multiple stances might suffer from the same concurrency issues we usually find in multithreaded applications.

Concurrency in GNUstep

What is Concurrency?

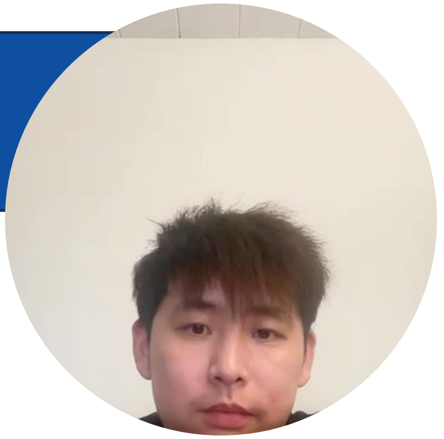
Why is Concurrency Needed in GNUstep?

Concurrency Mechanisms in GNUstep

Multi threading

Event-Driven Concurrency

Thread Synchronization



Control & Data Flow Summary

Front-end GUI processes user input

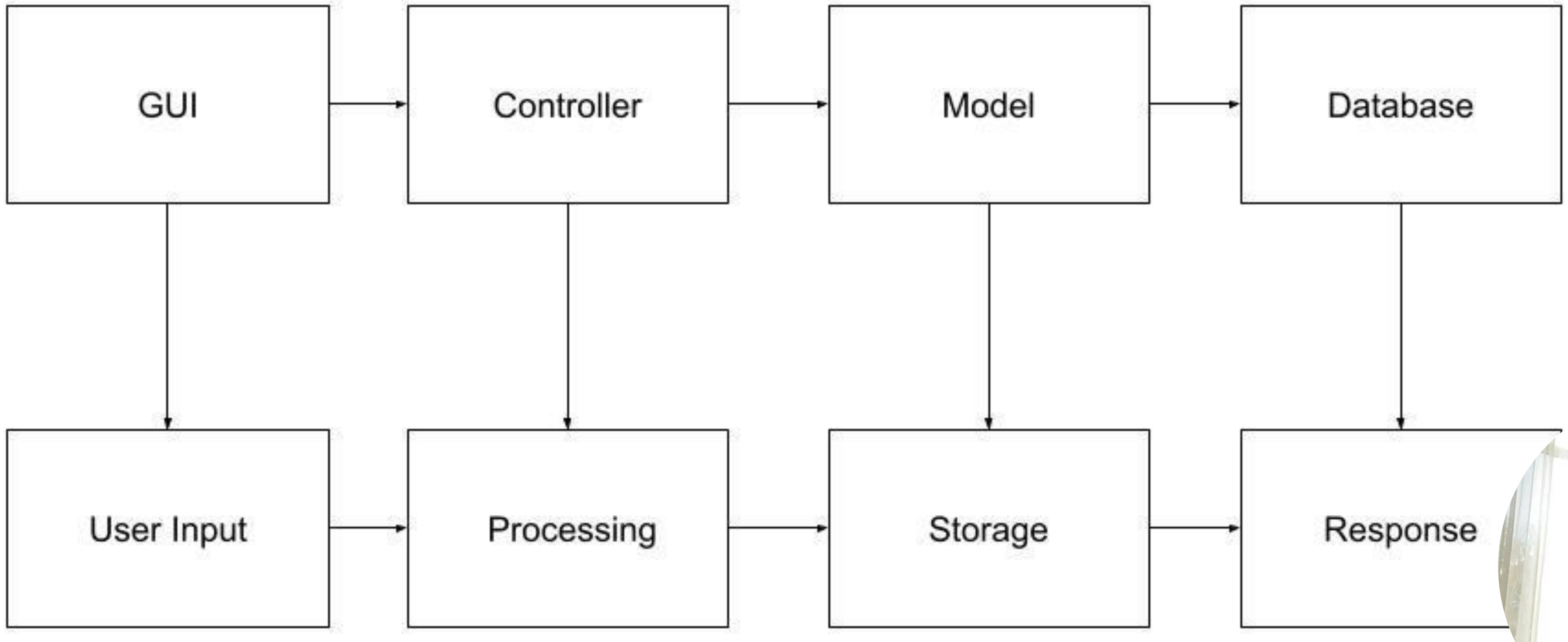
Application logic executes commands

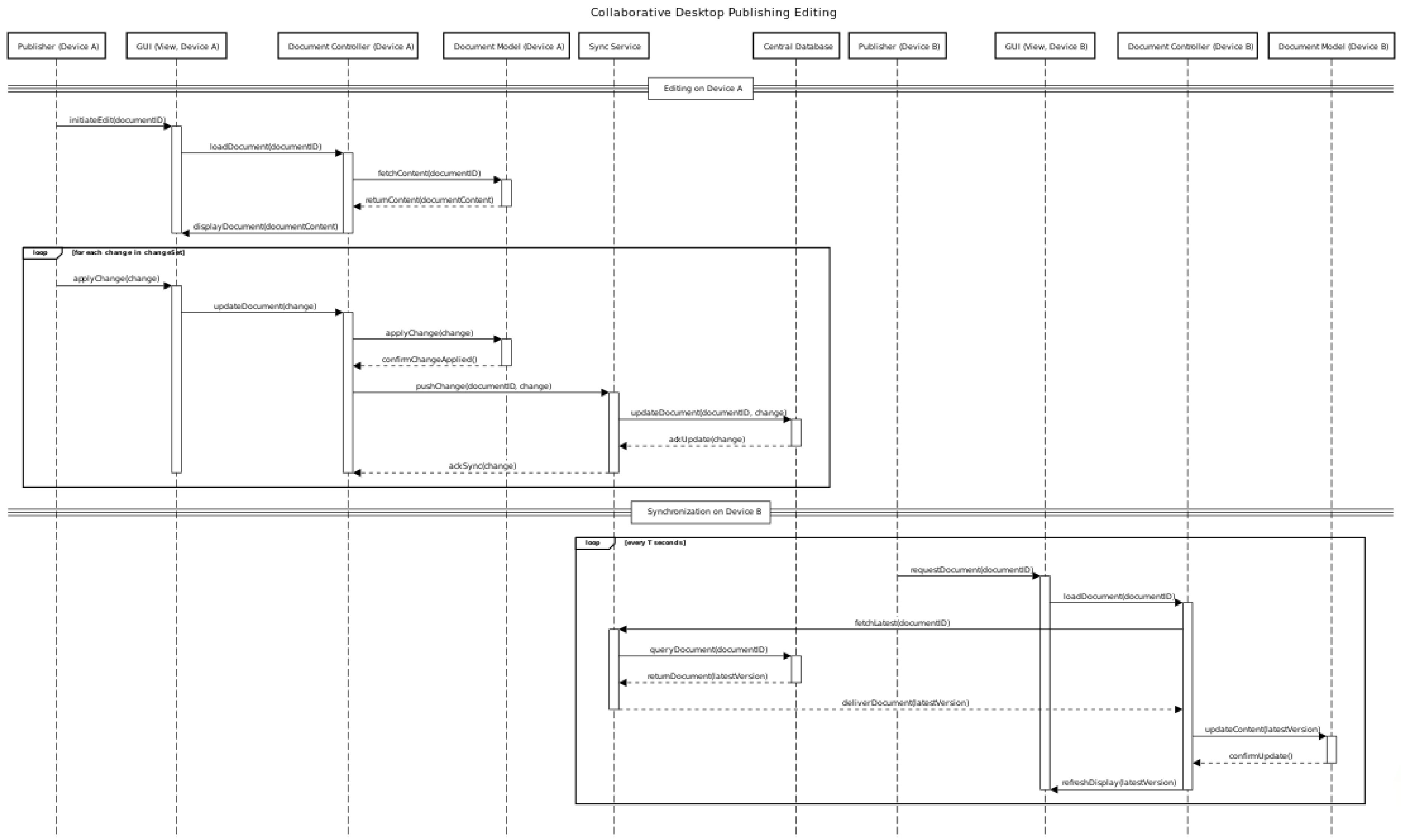
Middle-end handles cross-platform utilities

Back-end manages data and system resources

An event-driven model ensuring respon

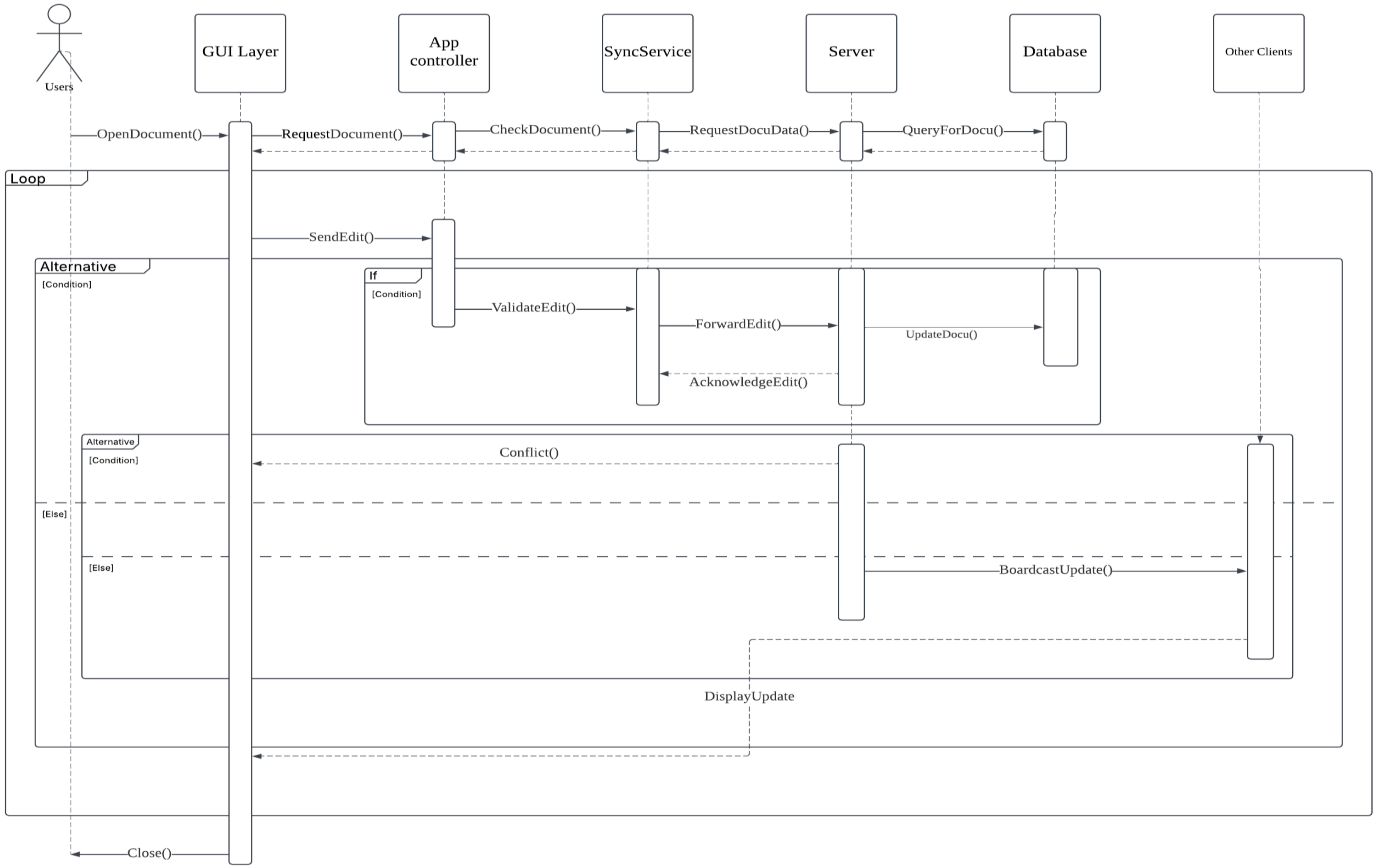
Box and Arrow Diagram





Collaborative Desktop Publishing Editing
Publisher (Device A)
GUI (View, Device A)
Document Controller (Device A)
Document Model (Device A)
Sync Service
Central Database
Publisher (Device B)
GUI (View, Device B)
Document Controller (Device B)
Document Model (Device B)
Editing on Device A
initiateEdit(documentID)
loadDocument(documentID)
fetchContent(documentID)
returnContent(documentContent)
displayDocument(documentContent)
loop
[for each change in changeSet]
applyChange(change)
updateDocument(change)
applyChange(change)
confirmChangeApplied()
pushChange(documentID, change)
updateDocument(documentID, change)
ackUpdate(change)
ackSync(change)
Synchronization on Device B
loop
[every T seconds]
requestDocument(documentID)
loadDocument(documentID)
fetchLatest(documentID)
queryDocument(documentID)
returnDocument(latestVersion)
deliverDocument(latestVersion)
updateContent(latestVersion)
confirmUpdate()
refreshDisplay(latestVersion)

Use Case 1: Multi-Device Application



Use Case 2: Collaborative Documents

Division of responsibilities among participating developers: independent or dependent?

- **Independent module:**
  - Foundation module (Libs-base)
  - File system
  - Data processing

- **Dependent module:**
  - User Interface (Libs-gui)
  - Rendering Backend (Libs-back)
  - Gorm

- Priority to independent modules, dependent modules build on top

Division of responsibilities: Cross-Team collaboration

- Clearly defined interfaces and API boundaries
- Branch-Based development
  - Code review and testing
- Regular meetings and syncs
- Automated testing framework

Lesson learned

- Cross-Platform Compatibility
- Layered Architecture
- Concurrency Management
- Control & Data Flow
- Division of Responsibility

Reference

1. GNUstep MediaWiki. Main page. Retrieved from https://mediawiki.gnustep.org/index.php/Main_Page
2. Clemson University. GNUstep manual. Retrieved from http://andrewd.ces.clemson.edu/courses/cpsc102/notes/GNUStep-manual.pdf
3. GNUstep. GUI Reference. Retrieved from https://www.gnustep.org/resources/documentation/Developer/Gui/Reference/index.html
4. GNUstep. Base Library Reference. Retrieved from https://www.gnustep.org/resources/documentation/Developer/Base/Reference/index.html
5. GNUstep. GNUstep Developer Documentation. Retrieved from https://www.gnustep.org/developers/documentation.html
6. GNUstep. GNUstep Base Library API Reference. Retrieved from https://www.gnustep.org/resources/documentation/Developer/Base/Reference/Base.html
7. GNUstep. NSThread Class Reference. Retrieved from https://www.gnustep.org/resources/documentation/Developer/Base/Reference/NSThread.html
8. GNUstep. NSLock Class Reference. Retrieved from https://www.gnustep.org/resources/documentation/Developer/Base/Reference/NSLock.html
9. GNUstepWiki. Foundation. Retrieved from https://mediawiki.gnustep.org/index.php/Foundation